

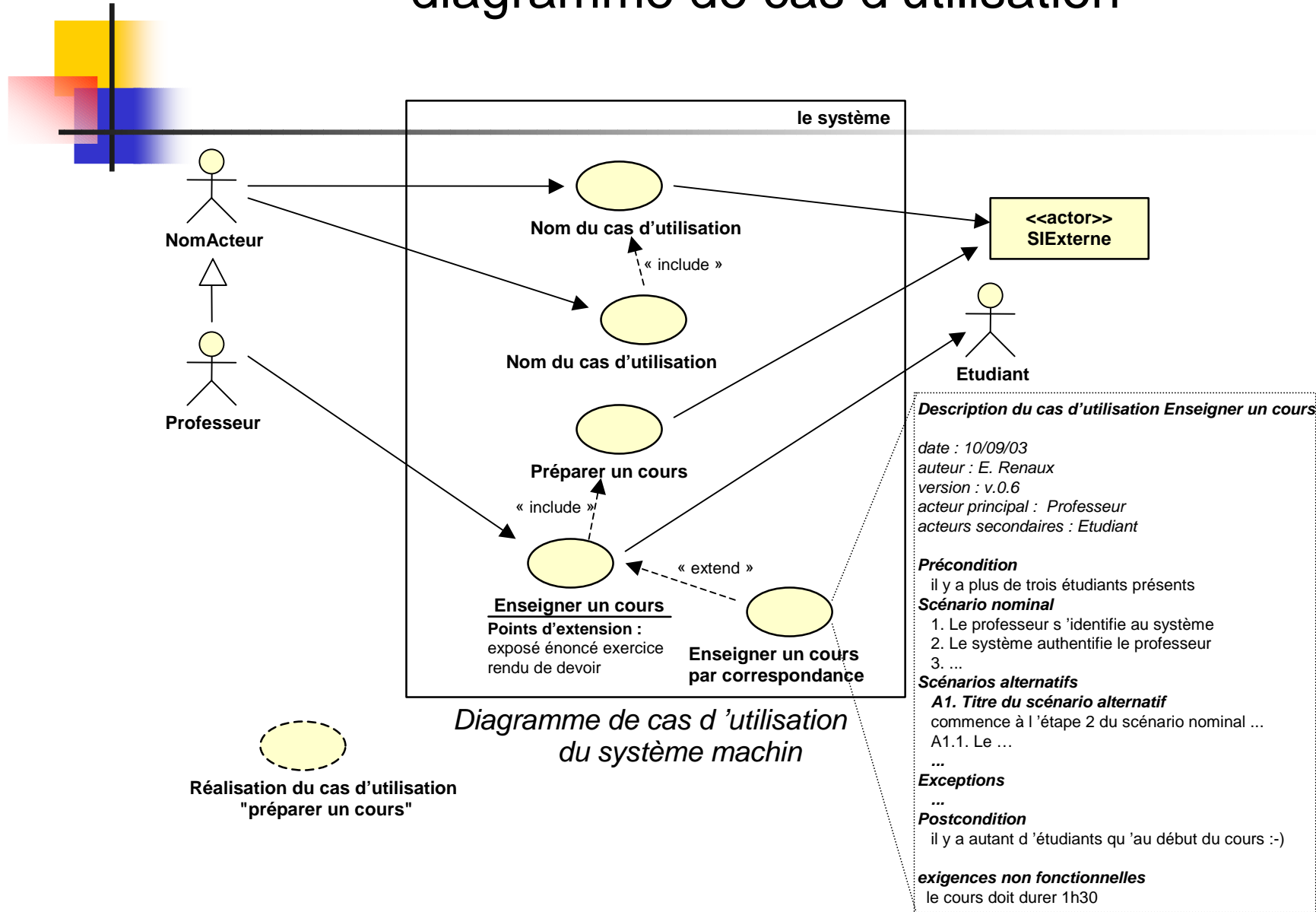
Introduction aux notations

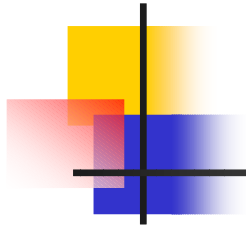
UML et OCL



use case diagram

diagramme de cas d'utilisation





activity diagram diagramme d'activités

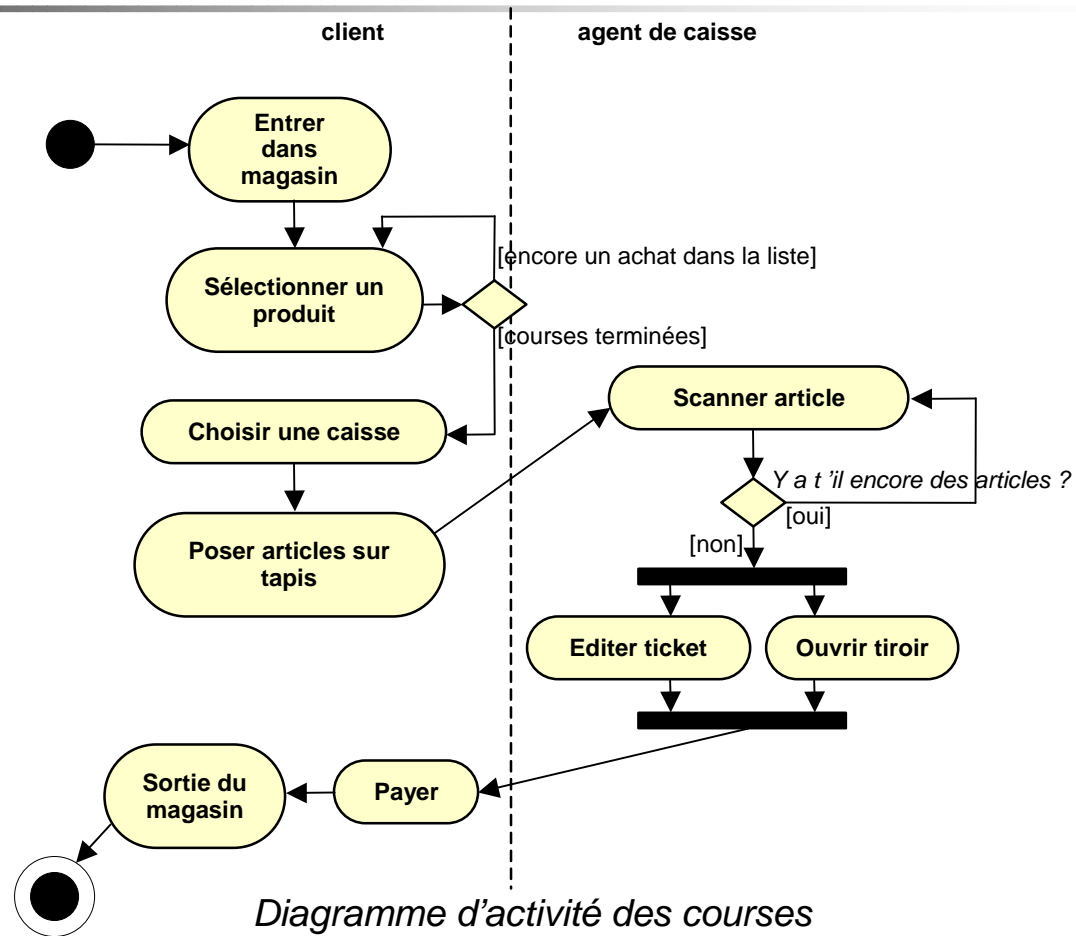


Diagramme d'activité des courses

class diagram diagramme de classes

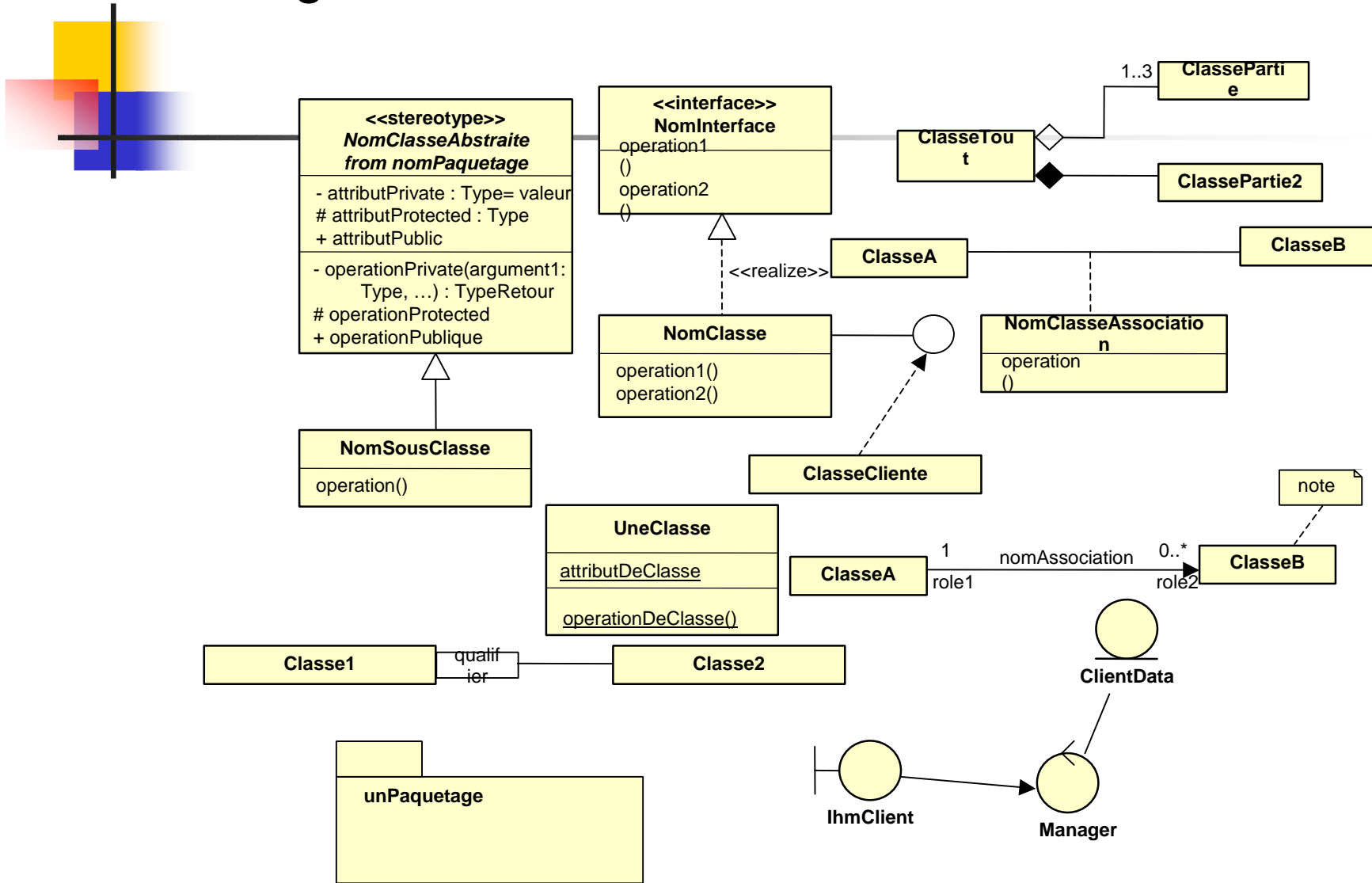


Diagramme de classe du système

sequence diagram diagramme de séquence

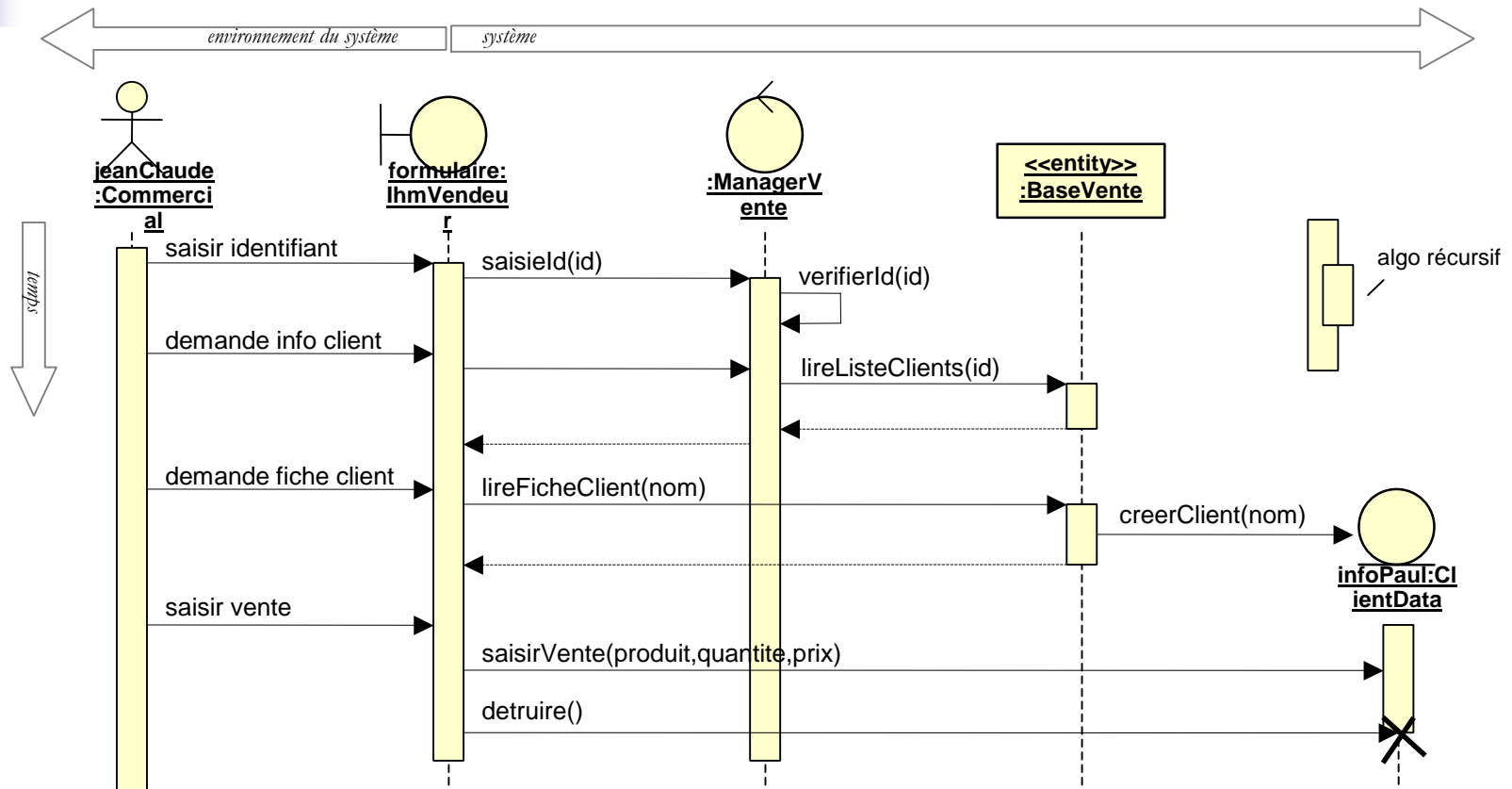


Diagramme de séquence du scénario nominal du cas d'utilisation "réaliser une vente"

statechart diagram
diagramme d'états

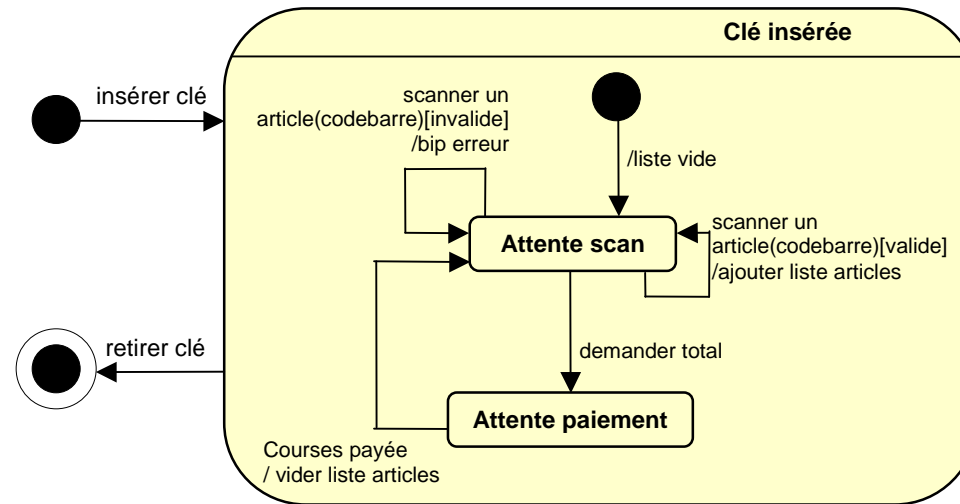


Diagramme d'états d'une caisse



component diagram
diagramme de composants

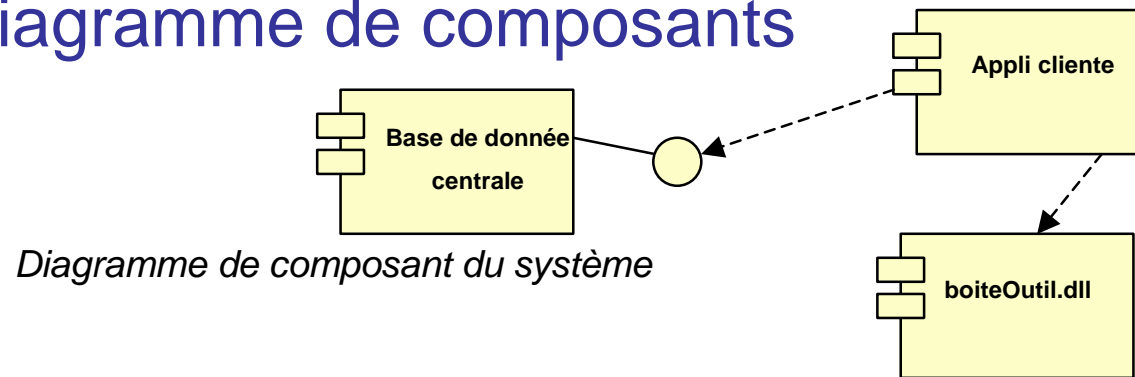
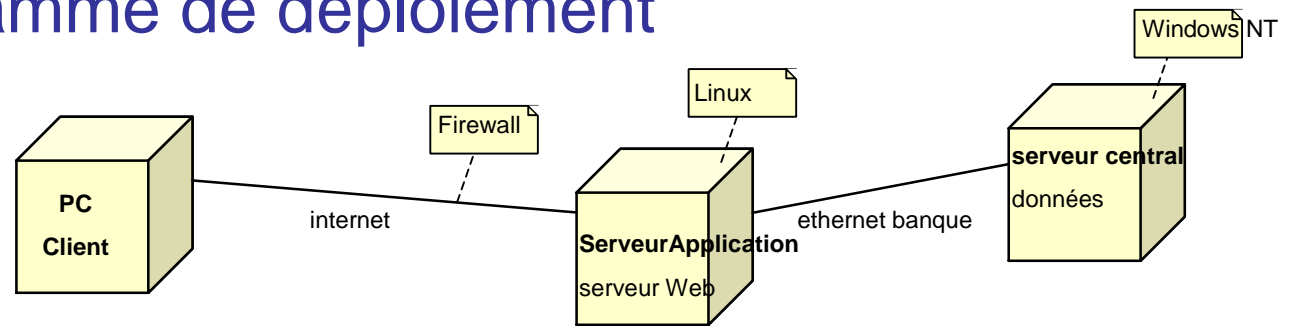


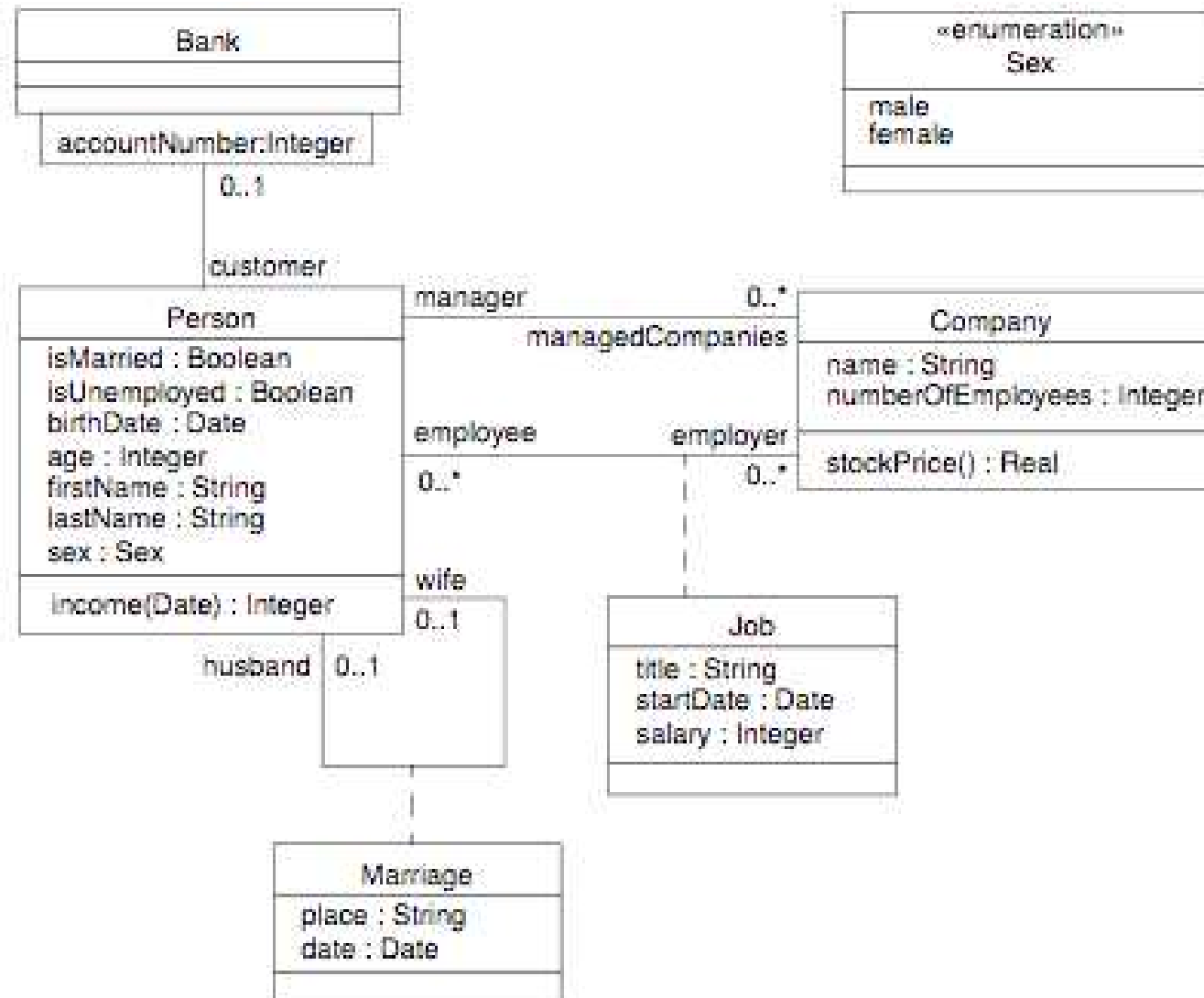
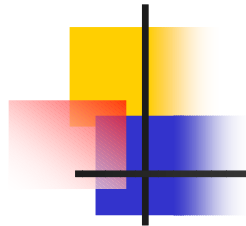
Diagramme de composant du système



deployment diagram
diagramme de déploiement



Limites d'UML





OCL

Object Constraint Language

d'après CSCI3007 Component Based Development



Plan

- Qu'est-ce qu'une contrainte?
- OCL Concepts
- OCL Concepts avancés



Definition

- “Une contrainte est la restriction de une ou plusieurs valeurs d’un modèle UML.”

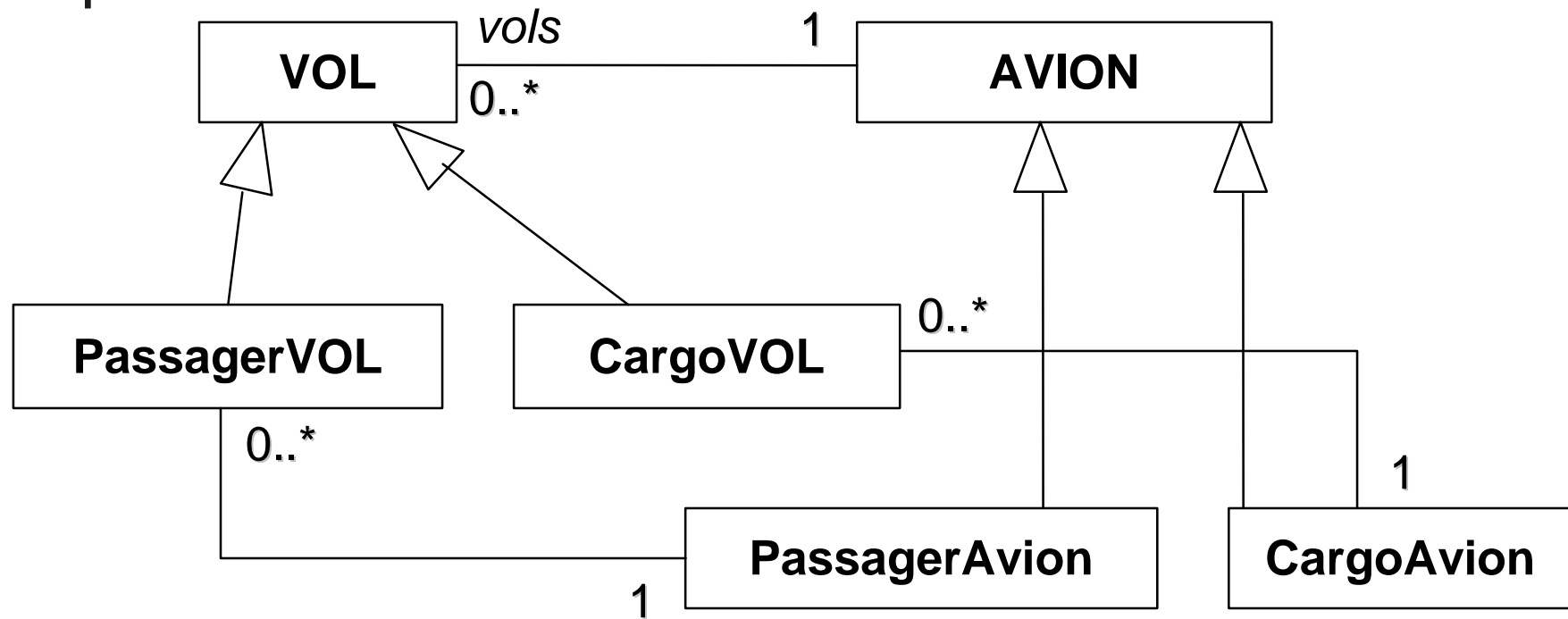


Différents types de contraintes

- Invariant de Classe
 - une contrainte qui doit être satisfaite par toutes les instances de la classe
- Precondition d'une operation
 - une contrainte qui doit toujours être *true* avant l'exécution de l' operation
- Postcondition d'une operation
 - une contrainte qui doit toujours être *true* après l'exécution de l' operation



préciser un diagramme





exemple: Ajouter des invariants



{context VOL

inv: type = #cargo implies avion.type = #cargo

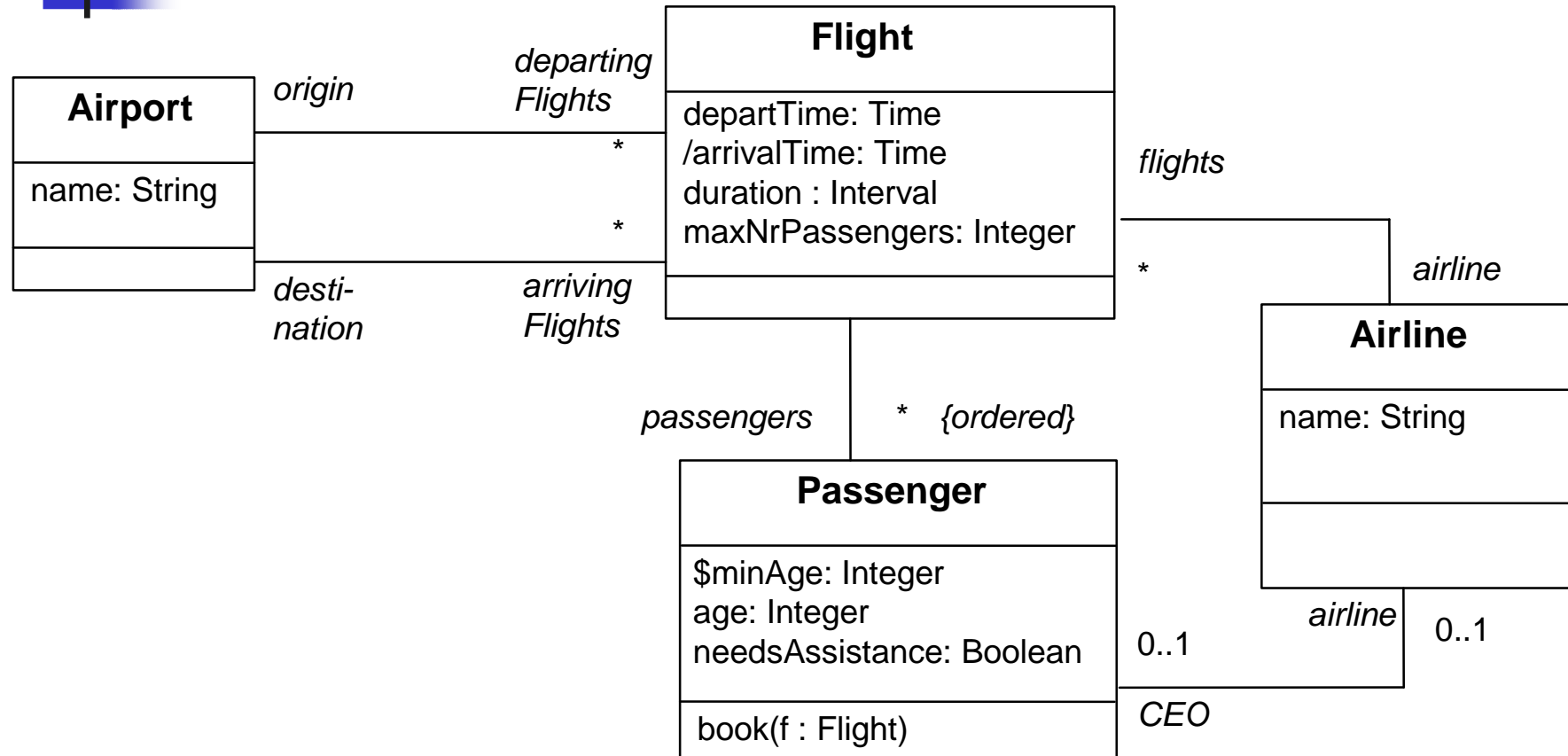
inv: type = #passager implies avion.type = #passager}



OCL?

- OCL est
 - un langage textuel pour décrire les contraintes dans UML
- Formel mais simple à utiliser
 - Non ambiguë
 - Sans effets de bord

Modèle Exemple





context et self

- toute expression OCL est associée à un contexte spécifique
 - Ce contexte est souvent l'élément UML auquel cette contrainte est attachée
- Ce contexte peut être désigné dans l'expression par le mot clé 'self'.
 - 'self' est implicite dans toute expression OCL
 - Équivalent à 'this' en Java

Notation

- Les Contraintes peuvent figurer dans le modèle UML ou dans un document séparé.

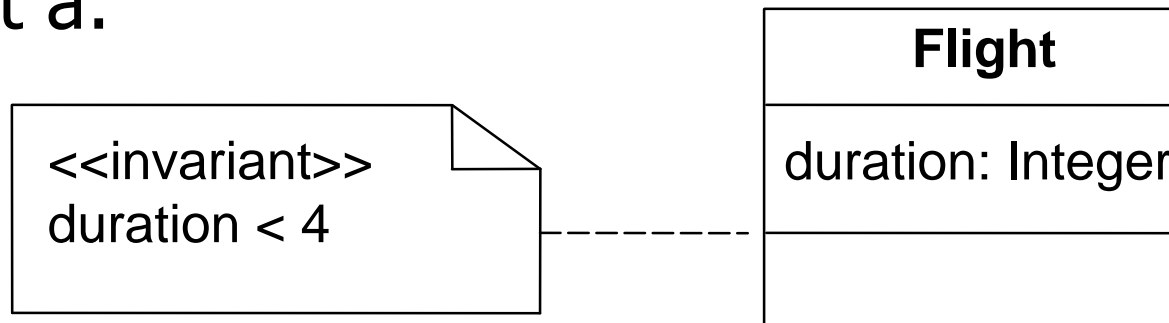
- L'expression:

context Flight inv: self.duration < 4

- Est identique à:

context Flight inv: duration < 4

- Et à:





Elements d'une expression OCL

- On peut utiliser:
 - Types de base: String, Boolean, Integer, Real.
 - Éléments du modèle UML
 - attributs, et classes
 - Operations booléennes des classes query operations (sans effets de bord)
 - Les associations du modèle UML
 - Y compris les noms de rôles
 - conseil: utiliser des rôles dans UML pour simplifier OCL



Exemple: OCL types de base

context Compagnie inv:

name.toLower = **'klm'**

context Passenger inv:

age $\geq ((9.6 - 3.5) * 3.1)$.floor implies

mature = **true**



Classes, Modèle et attributs

- attributs

context VOL inv:

self.**maxNrPassagers** <= 1000

- Attributs de Classe

context Passenger inv:

age >= Passenger.**minAge**



Exemple: operations

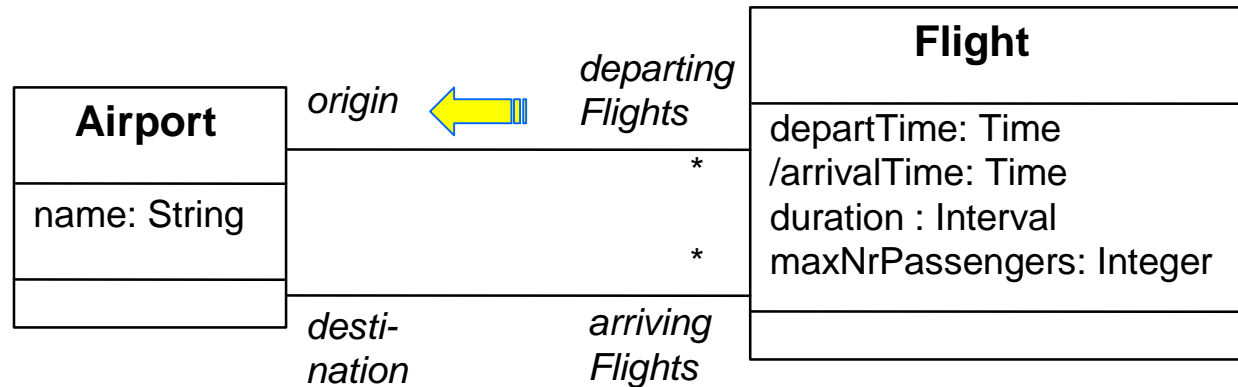
context Flight inv:

```
self.departTime.difference(self.arrivalTime)  
  .equals(self.duration)
```

Time
<u>midnight</u> : Time month : String day : Integer year : Integer hour : Integer minute : Integer
difference(t:Time):Interval before(t: Time): Boolean plus(d : Interval) : Time

Interval
nrOfDays : Integer nrOfHours : Integer nrOfMinutes : Integer
equals(i:Interval):Boolean <u>Interval(d, h, m : Integer) :</u> <u>Interval</u>

Exemple: navigations



context Flight

inv: origin <> destination

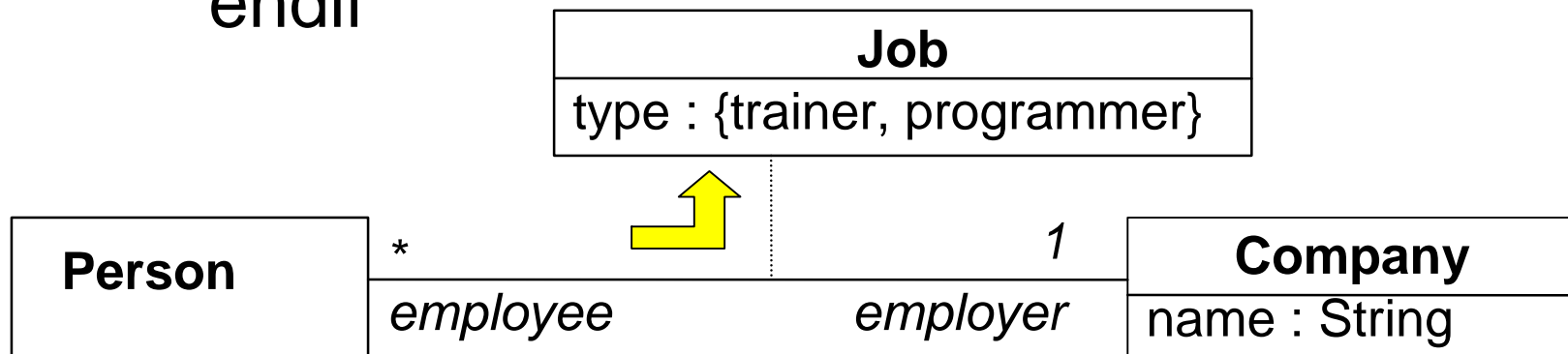
inv: origin.name = 'Amsterdam'

context Flight

inv: airline.name = 'KLM'

Classes Association

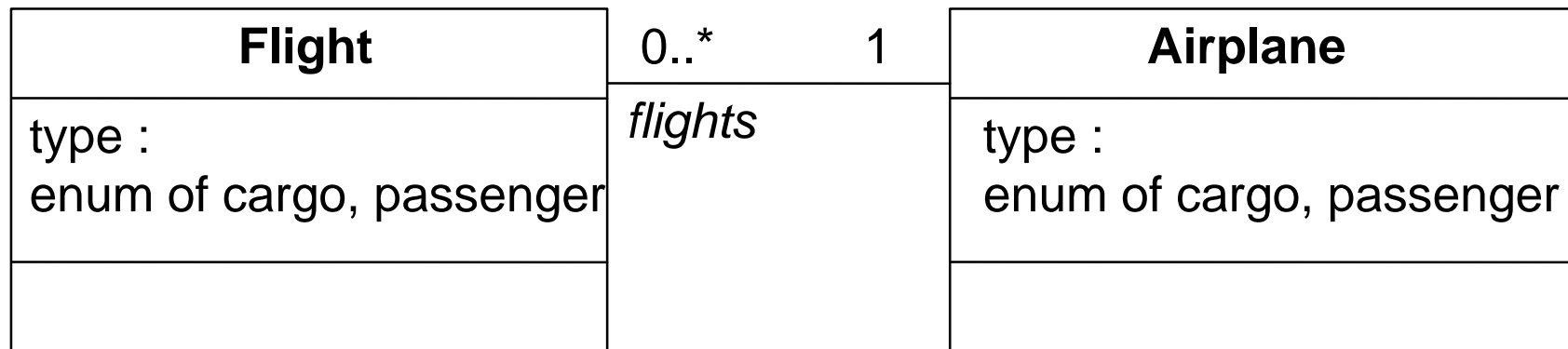
```
context Person inv:  
if employer.name = 'Klasse Objecten' then  
  job.type = #trainer  
else  
  job.type = #programmer  
endif
```





Collections dans OCL


- La plupart des navigations retournent des collections d'elements





Trois Sous types de Collection

- Set:
 - arrivingFlights(dans le contexte Airport)
 - Non-ordonné, unique
- Bag:
 - arrivingFlights.duration (dans le contexte Airport)
 - Non-ordonné, non-unique
- Sequence:
 - passagers (dans le contexte Flight)
 - ordonné, non-unique



Collection : opérations

- OCL a un grand nombre d'opérations prédéfinies sur les collections.
- Syntax:
 - collection -> operation



collect

- Syntaxe:

 - collection->collect(elem : T | expr)

 - collection->collect(elem | expr)

 - collection->collect(expr)

- résumé:

 - collection.expr

- L'opération *collect* retourne la collection des valeurs obtenues par application de *expr* aux éléments de *collection*



select

- **Syntaxe:**

- collection->select(elem : T | expression)

- collection->select(elem | expression)

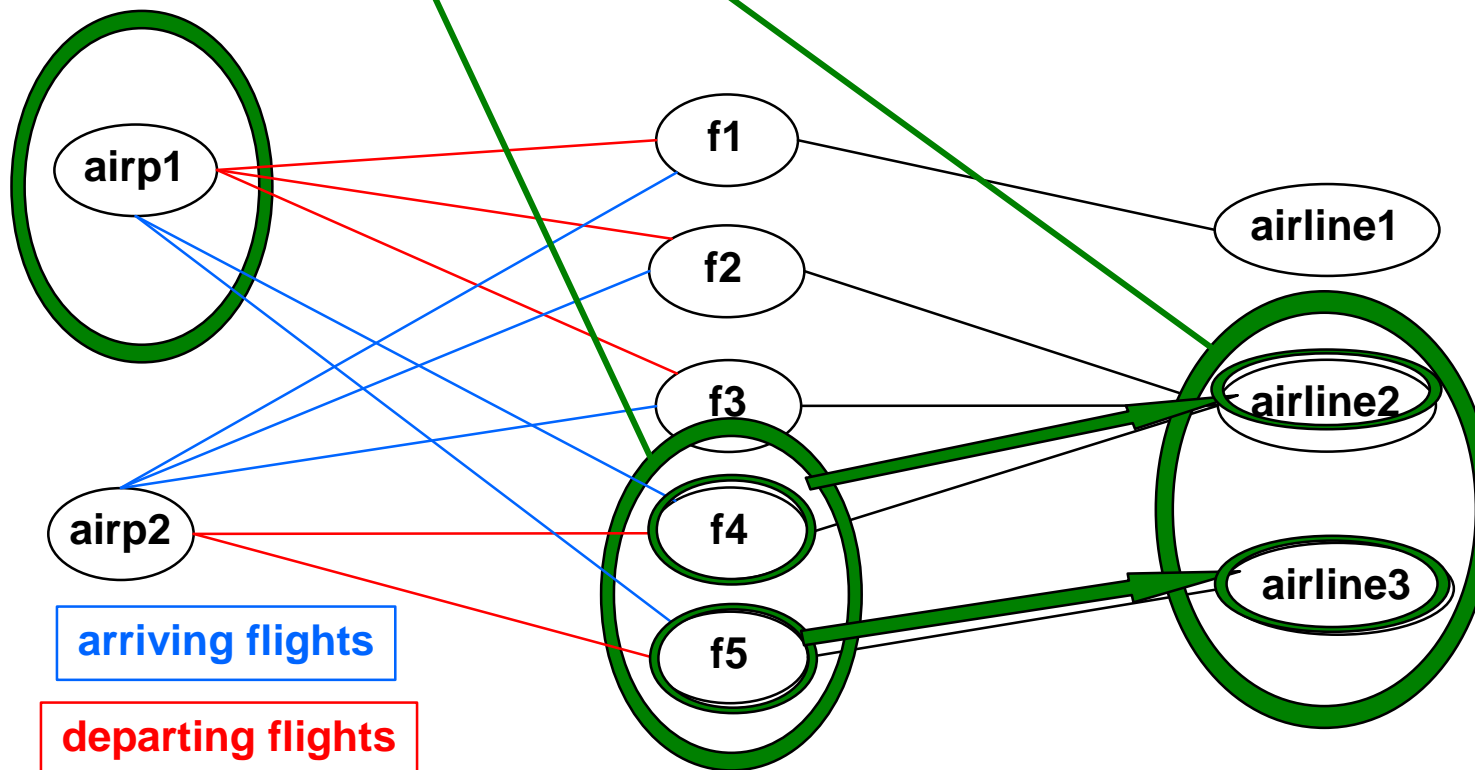
- collection->select(expression)

- *select retourne le sous ensemble des éléments pour lesquels expression est true*

Exemple: collect

context Airport inv:

`self.arrivingFlights` -> `collect(airLine)` -> `notEmpty`





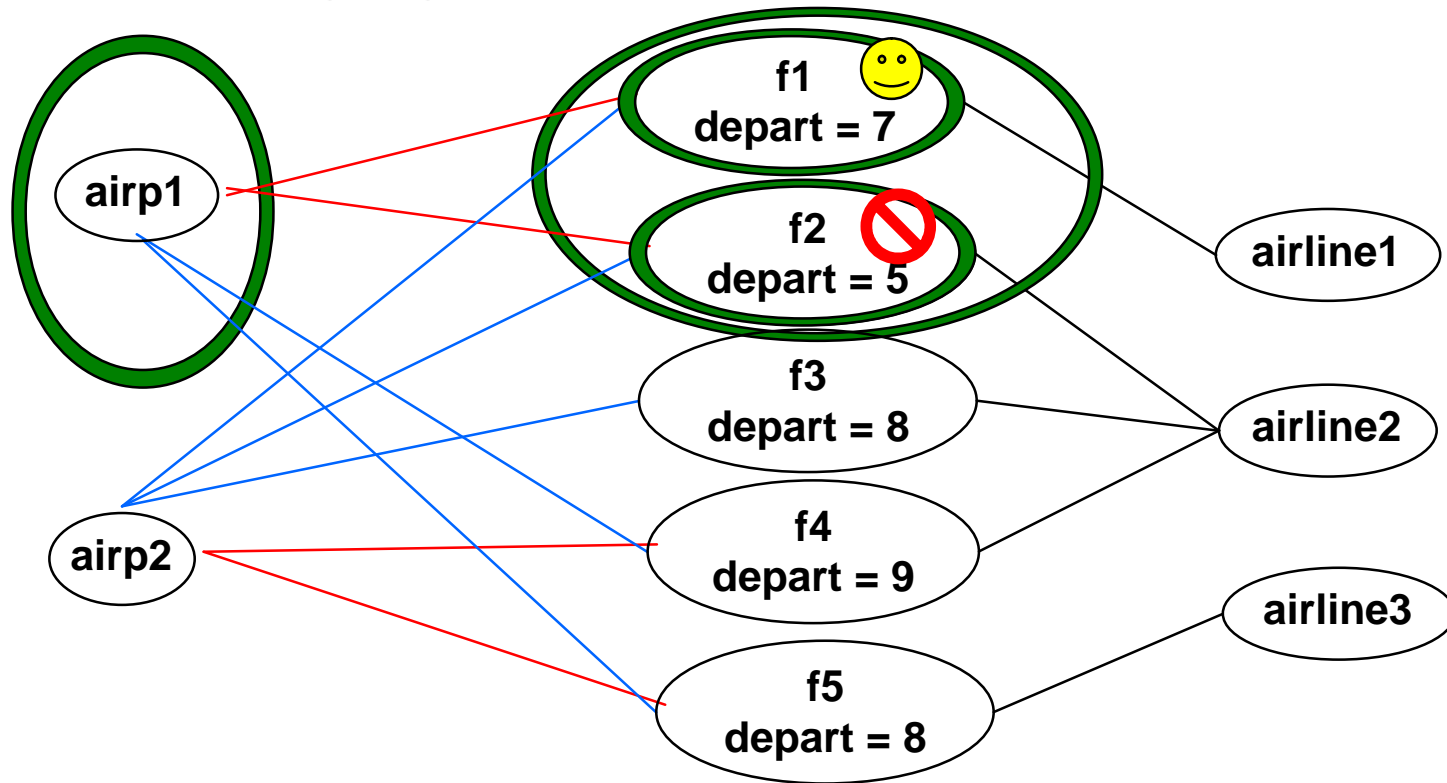
forall

- Syntaxe:
 - `collection->forall(elem : T | expr)`
 - `collection->forall(elem | expr)`
 - `collection->forall(expr)`
- *True* si *expr* est *true* pour tous les éléments de *collection*

Exemple: forAll

context Airport inv:

self.departingFlights->forAll(departTime.hour>6)



departing flights

arriving flights



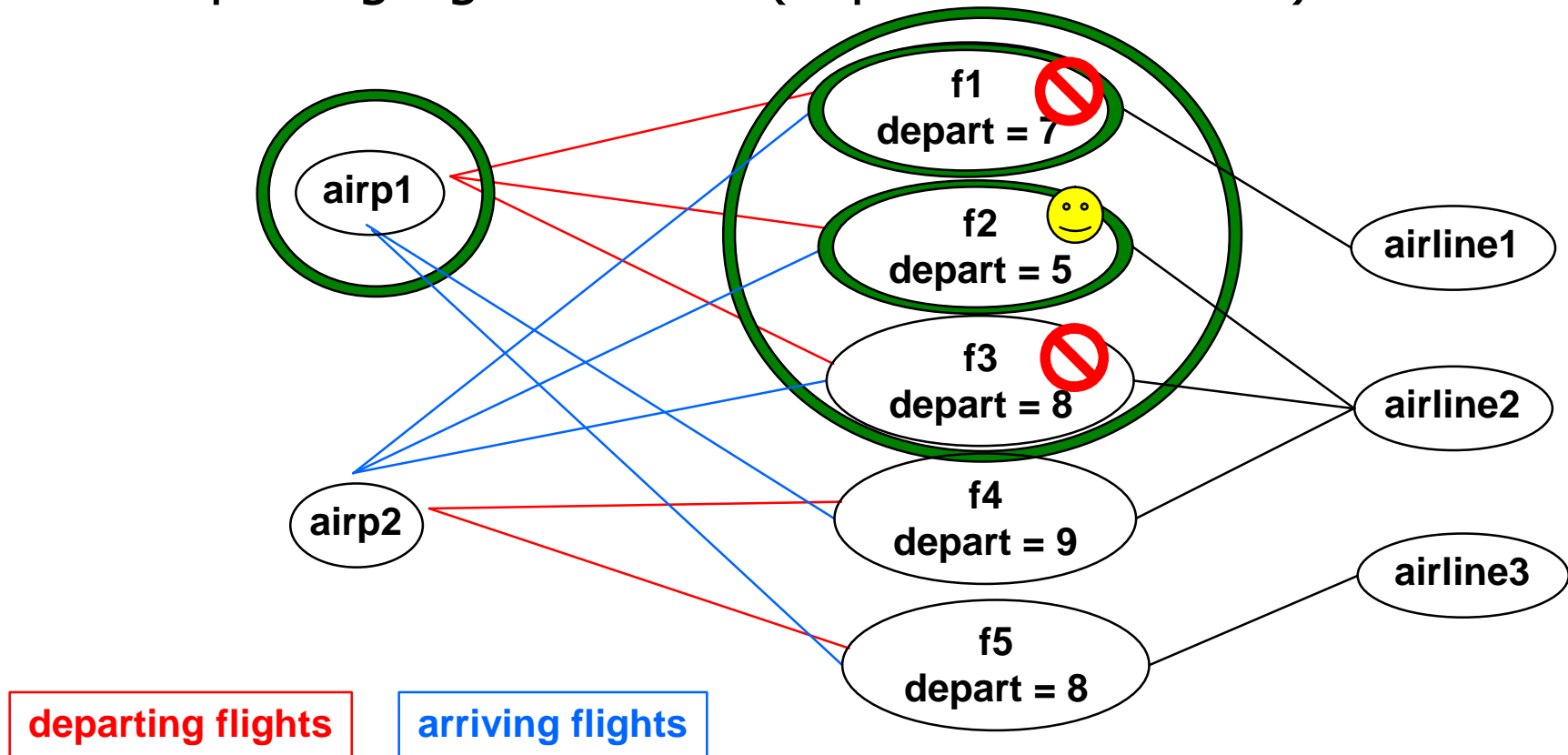
exists

- **Syntaxe:**
 - collection->exists(elem : T | expr)
 - collection->exists(elem | expr)
 - collection->exists(expr)
- *true* si il existe au moins un élément de *collection* pour lequel *expr est true*.

Exemple: exists

context Airport inv:

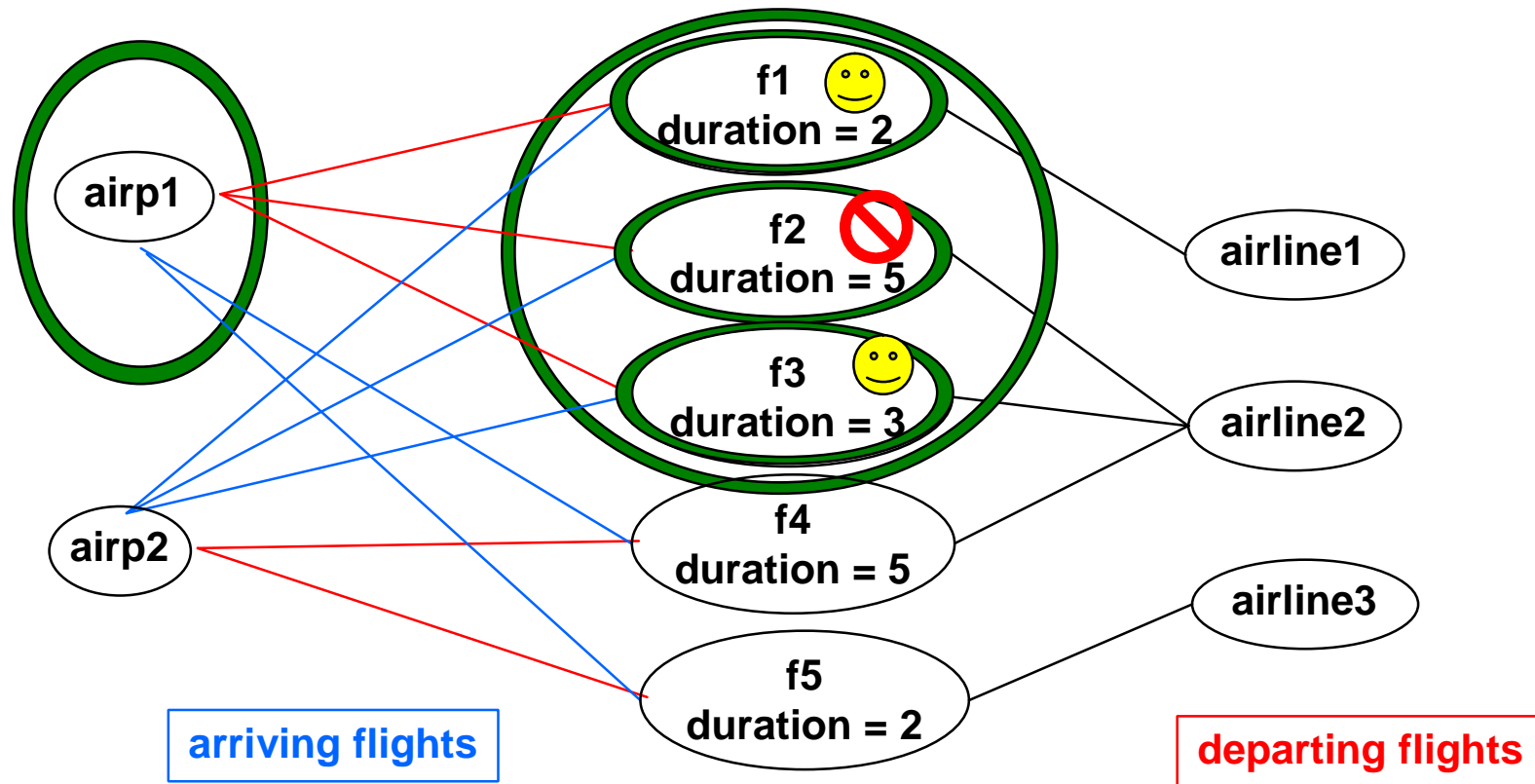
self.departingFlights->exists(departTime.hour<6)



Exemple: select

context Airport inv:

self.departingFlights->select(duration<4)->notEmpty





Exemple : Iterate

- Exemple iterate:

context Airline inv:

flights->select(maxNrPassengers > 150)->notEmpty

- Est identique à:

context Airline inv:

flights->iterate (f : Flight;

 answer : Set(Flight) = Set{ } |

 if f.maxNrPassengers > 150 then

 answer->including(f)

 else

 answer endif)->notEmpty



autres operations sur collection

- *isEmpty*
- *notEmpty*
- *size*
- *count(elem)*: occurrences d'elem dans collection
- *includes(elem)*: true si elem dans collection
- *excludes(elem)*
- *includesAll(coll)*



Result et postcondition

- Exemple pre et postcondition

```
context Airline::servedAirports() : Set(Airport)
```

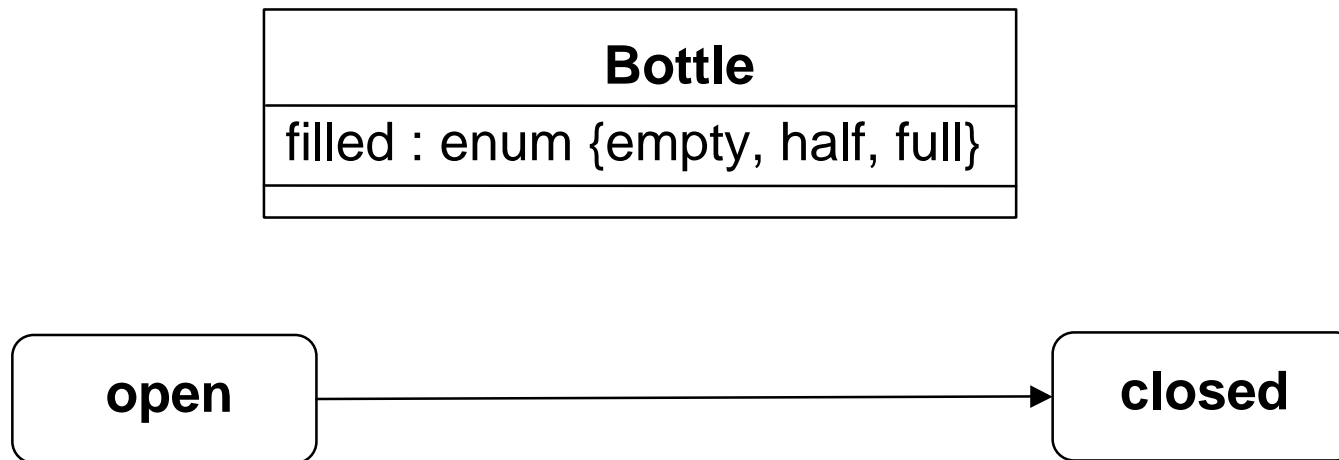
```
pre : -- none
```

```
post: result = flights.destination->asSet
```




Statechart

- L' operation *oclInState* retourne true si l'objet est dans l'état spécifié



context Bottle inv:

self.**oclInState(closed)** implies filled = #full



Variables Locales

- *let* définit des variables locales à une contrainte:

Let var : Type = <expression1> in
 <expression2>

- Exemple:

context Airport inv:

Let **supportedAirlines** : Set (Airline) =
 self.arrivingFlights -> collect(airLine) in
 (**supportedAirlines** ->notEmpty) and
 (**supportedAirlines** ->size < 500)



heritage de contraintes

- Liskov's Substitution Principle (LSP):
 - "Partout où une instance d'une classe est attendue, une instance d'une sous classe peut lui être substituée"



heritage de contraintes

- Consequences de LSP pour les invariants:
 - un invariant est toujours hérité par ses sous classes.
 - Les sousclasses peuvent renforcer l'invariant.
- Consequences de LSP sur les preconditions et postconditions:
 - une precondition peut être weakened (contravariance)
 - une postcondition peut être strengthened (covariance)



OCL Tools

- FREE parser from IBM
 - <http://www.software.ibm.com/ad/ocl>
- Cybernetics
 - www.cybernetic.org
- University of Dresden
 - www-st.inf.tu-dresden.de/ocl/
- Boldsoft
 - www.boldsoft.com
- ICON computing
 - www.iconcomp.com
- Royal Dutch Navy
- Others



Conclusions

- OCL invariants allow you to
 - model more precisely
 - remain implementation independent
- OCL pre- and postconditions allow you to
 - specify contracts (design by contract)
 - specify interfaces of components more precisely
- OCL usage tips
 - keep constraints simple
 - always combine natural language with OCL
 - use a tool to check your OCL

References



- [UML 1.3] *OMG UML Specification v. 1.3*,
OMG doc# ad/06-08-99
- [UML 1.4] *OMG UML Specification v. 1.4*, UML
Revision Task Force recommended final draft,
OMG doc# ad/01-02-13.



Infos

- Web:
 - UML 1.4 RTF: www.celigent.com/omg/umlrtf
 - OMG UML Tutorials: www.celigent.com/omg/umlrtf/tutorials.htm
 - UML 2.0 Working Group: www.celigent.com/omg/adptf/wgs/uml2wg.htm
 - OMG UML Resources: www.omg.org/uml/